Car Purchase Prediction Model

Karan Prajapati

Data Pre-Processing

```r
# List for Storing performance metrics
metrics <- list()

# Importing data set
dataset <- read.csv('car_data.csv')
dataset <- dataset[, 3:5]

# Encoding the target feature as factor
dataset$Purchased <- factor(dataset$Purchased, levels = c(0, 1))

# Run this if the to install the required packages
# install.packages("caTools", "e1071", "caret", "class", "rpart", "randomForest")
# or install from the RStudio install tab

# library imports
library(caTools) # used for initial data pre-processing

# Libraries for models
library(class) # dependency of all classification models
library(e1071) # used for svm,
library(caret) # used for performance metrics
library(rpart) # used for decision tree
library(randomForest) # used for random forest

# Libraries for plotting
library(ggplot2) # used for plotting graphs
library(tidyr) # used for reshaping data

# Setting the values to not be truly random
set.seed(123)

# Split data into 80% Training Set and 20% Test Set
split <- sample.split(dataset$Purchased, SplitRatio = 0.80)

training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)

# Feature Scaling
training_set[, 1:2] <- scale(training_set[, 1:2])
test_set[, 1:2] <- scale(test_set[, 1:2])
```

Calculating the performance metrics of our prediction

```
get_accuracy <- function(training_data, prediction, name) {
  prediction_factor <- factor(prediction, levels = c(0, 1))
  actual_factor <- factor(training_data$Purchased, levels = c(0, 1))

  cm <- confusionMatrix(prediction_factor, actual_factor)

  cat("\n")
  print(name)
  cat("\n")

  print(cm)
  return (cm)
}
```

Create a grid to plot the graph

```
get_grid_set <- function() {
  set = test_set

  # Define a range for the grid
  X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
  X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

  grid_set <- expand.grid(X1, X2) # create a grid
  colnames(grid_set) <- c("Age", "AnnualSalary") # name the grid lines
  return (list(grid = grid_set, x1 = X1, x2 = X2))
}
```

Function to plot the graph of prediction

```
plot_graph <- function(prediction, grid_set, name) {
  set = test_set

  grid <- grid_set$grid # plane of the plot
  X1 <- grid_set$x1 # X axis
  X2 <- grid_set$x2 # Y axis

  # Plot Graph
  plot(set[, -3],
       main = name,
       xlab = "Age", ylab = "Annual Salary",
       xlim = range(X1), ylim = range(X2))

  # Decision Boundary
  contour(X1, X2, matrix(as.numeric(prediction), length(X1), length(X2)), add = TRUE)
```

```r
  # Color the regions
  points(grid, pch = '.', col = ifelse(prediction == 1, 'springgreen3', 'tomato'))

  # Plot the actual test points
  points(set, pch = 21, bg = ifelse(set[, 3] == 1, "green4", "red3"))
}
```

Append data to the metrics list

```r
set_metrics <- function(cm, key) {
  f1_score <- as.numeric(cm$byClass['F1'])
  accuracy <- as.numeric(cm$overall['Accuracy'])

  metrics[[key]] <<- list(f1 = f1_score, acc = accuracy)
}
```

Display the performance metrics and plot the graph

```r
show_results <- function(y_pred_normal, y_pred_plot, normal_name) {
  cm <- get_accuracy(test_set, y_pred_normal, normal_name)
  set_metrics(cm, normal_name)
  plot_graph(y_pred_plot, get_grid_set(), paste(normal_name, "Plotting"))
}
```

Logistic Regression Implementation

```r
logistic_regression_pred <- function() {
  classifier <- glm(formula = Purchased ~ .,
                    family = binomial,
                    data = training_set)

  # Prediction for confusion matrix
  prob_pred_normal = predict(classifier, type = "response", newdata = test_set[-3])
  y_pred_normal = ifelse(prob_pred_normal > 0.5, 1, 0) # 50% threshold value

  # Prediction for visualization
  prob_pred_plot = predict(classifier, type = "response", newdata = get_grid_set()$grid)
  y_pred_plot = ifelse(prob_pred_plot > 0.5, 1, 0) # 50% threshold value

  # Display the performance metrics and plot the graph
  show_results(y_pred_normal, y_pred_plot, "Logistic Regression")
}
```

Support Vector Machine Implementation

```
svm_pred <- function() {
  classifier <- svm(formula = Purchased ~ .,
                    data = training_set,
                    type = 'C-classification',
                    kernel = 'linear')

  # Prediction for confusion matrix
  y_pred_normal = predict(classifier, newdata = test_set[-3])

  # Prediction for visualization
  y_pred_plot = predict(classifier, newdata = get_grid_set()$grid)

  # Display the performance metrics and plot the graph
  show_results(y_pred_normal, y_pred_plot, "Support Vector Machine (SVM)")
}
```

Decision Tree Implementation

```
decision_tree_pred <- function() {
  classifier <- rpart(formula = Purchased ~ .,
                      data = training_set)

  # Prediction for confusion matrix
  y_pred_normal = predict(classifier, newdata = test_set[-3], type = 'class')

  # Prediction for visualization
  y_pred_plot = predict(classifier, newdata = get_grid_set()$grid, type = 'class')

  # Display the performance metrics and plot the graph
  show_results(y_pred_normal, y_pred_plot, "Decision Tree")
}
```

Random Forest Implementation

```
random_forest_pred <- function() {
  classifier <- randomForest(x = training_set[-3],
                             y = training_set$Purchased,
                             ntree = 500)

  # Prediction for confusion matrix
  y_pred_normal = predict(classifier, newdata = test_set[-3])

  # Prediction for visualization
  y_pred_plot = predict(classifier, newdata = get_grid_set()$grid)
```

```r
  # Display the performance metrics and plot the graph
  show_results(y_pred_normal, y_pred_plot, "Random Forest")
}
```

K-Nearest Neighbors Implementation

```r
knn_pred <- function() {
  # Prediction for confusion matrix
  y_pred_normal <- knn(train = training_set[, -3],
                       test = test_set[-3],
                       cl = training_set[, 3],
                       k = 5)

  # Prediction for visualization
  y_pred_plot <- knn(train = training_set[, -3],
                     test = get_grid_set()$grid,
                     cl = training_set[, 3],
                     k = 5)

  # Display the performance metrics and plot the graph
  show_results(y_pred_normal, y_pred_plot, "K-Nearest Neighbors (KNN)")
}
```

Convert the metrics list to a data frame

```r
get_metrics_df <- function() {
  # Define a data frame for plotting
  metrics_df <- data.frame(
    model = character(),
    variable = character(),
    value = numeric(),
    stringsAsFactors = FALSE
  )

  # Create a new row with the model names
  for (model in names(metrics)) {
    metrics_df <- rbind(metrics_df,
                        data.frame(
                          model = model,
                          variable = "Accuracy",
                          value = metrics[[model]]$acc
                        ))

    metrics_df <- rbind(metrics_df,
                        data.frame(
                          model = model,
                          variable = "F1-Score",
                          value = metrics[[model]]$f1
```

```
                         ))
  }
  return (metrics_df)
}
```

Draw the bar chart

```
draw_bar_chart <- function(name) {
  metrics_df = get_metrics_df()
  ggplot(metrics_df[metrics_df$variable == name, ],
         aes(x = model, y = value, fill = model)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = round(value, 2)), vjust = -0.3, size = 3) +
    labs(title = paste("Model", name, "Comparision"), x = "Model", y = name) +
    theme_minimal() +
    theme(axis.text = element_text(angle = 45, hjust = 1))
}
```

Logistic Regression Prediction

```
logistic_regression_pred()
```

```
##
## [1] "Logistic Regression"
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 104  22
##          1  16  58
##
##                Accuracy : 0.81
##                  95% CI : (0.7487, 0.8619)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 1.623e-10
##
##                   Kappa : 0.5992
##
##  Mcnemar's Test P-Value : 0.4173
##
##             Sensitivity : 0.8667
##             Specificity : 0.7250
##          Pos Pred Value : 0.8254
##          Neg Pred Value : 0.7838
##              Prevalence : 0.6000
##          Detection Rate : 0.5200
```

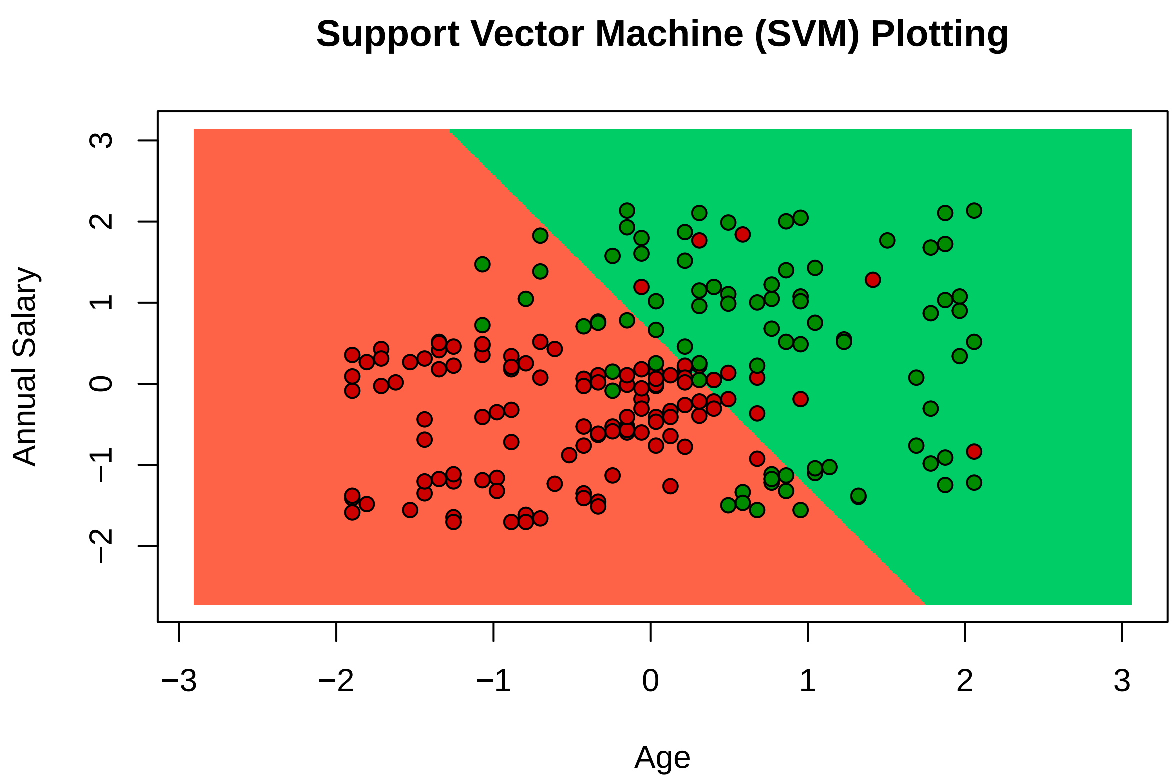
```
##    Detection Prevalence : 0.6300
##       Balanced Accuracy : 0.7958
##
##        'Positive' Class : 0
##
```

Support Vector Machine (SVM) Prediction

```r
svm_pred()
```

```
##
## [1] "Support Vector Machine (SVM)"
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 105  23
##          1  15  57
##
##                Accuracy : 0.81
##                  95% CI : (0.7487, 0.8619)
##     No Information Rate : 0.6
```

```
##     P-Value [Acc > NIR] : 1.623e-10
##
##                   Kappa : 0.5975
##
##  Mcnemar's Test P-Value : 0.2561
##
##             Sensitivity : 0.8750
##             Specificity : 0.7125
##          Pos Pred Value : 0.8203
##          Neg Pred Value : 0.7917
##              Prevalence : 0.6000
##          Detection Rate : 0.5250
##    Detection Prevalence : 0.6400
##       Balanced Accuracy : 0.7937
##
##        'Positive' Class : 0
##
```

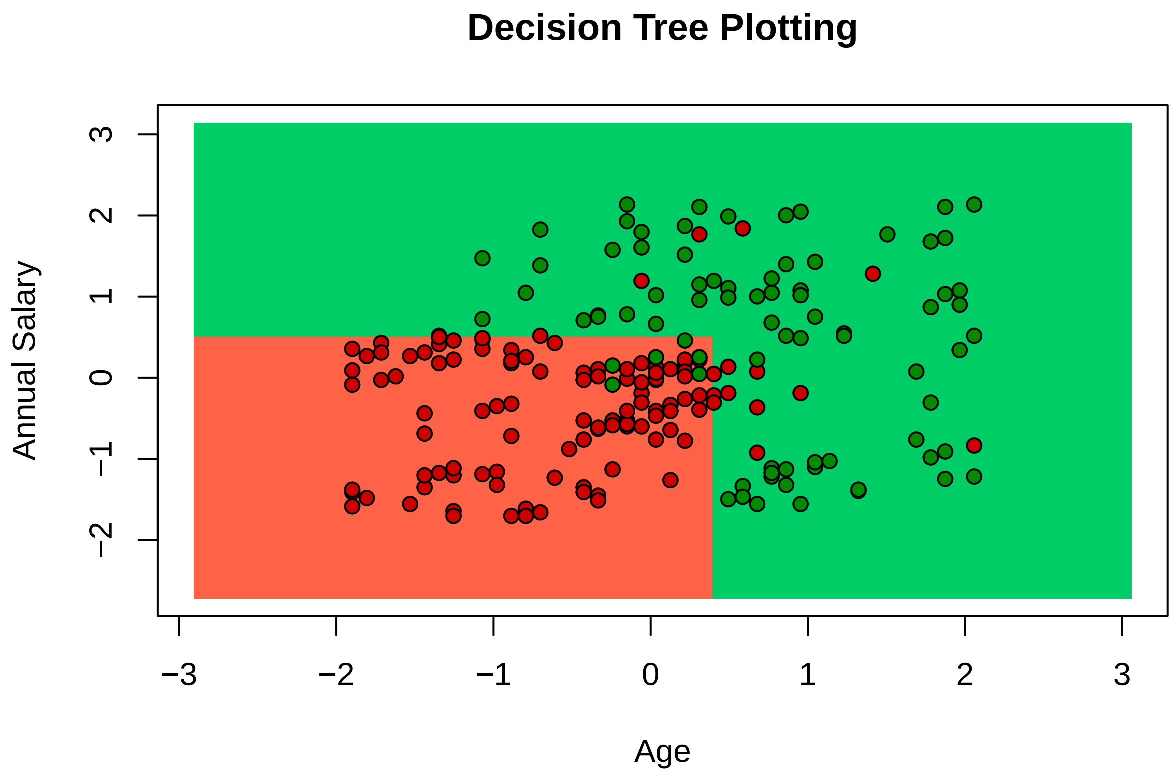


Decision Tree Prediction

```
decision_tree_pred()
```

```
##
```

```
## [1] "Decision Tree"
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 104   6
##          1  16  74
##
##                Accuracy : 0.89
##                  95% CI : (0.8382, 0.9298)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7755
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.8667
##             Specificity : 0.9250
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.8222
##              Prevalence : 0.6000
##          Detection Rate : 0.5200
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.8958
##
##        'Positive' Class : 0
##
```

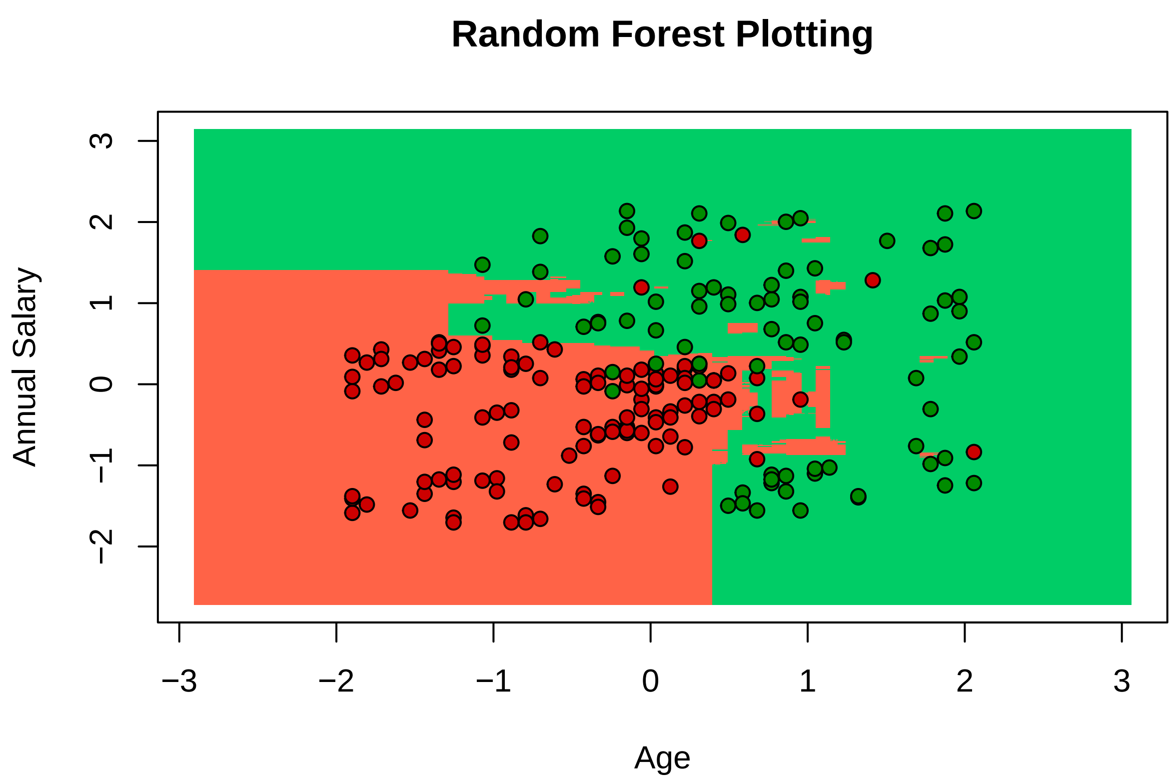


Random Forest Prediction

```
random_forest_pred()
```

```
##
## [1] "Random Forest"
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 111   8
##          1   9  72
##
##                Accuracy : 0.915
##                  95% CI : (0.8674, 0.9497)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8233
##
##  Mcnemar's Test P-Value : 1
##
```

```
##             Sensitivity : 0.9250
##             Specificity : 0.9000
##          Pos Pred Value : 0.9328
##          Neg Pred Value : 0.8889
##              Prevalence : 0.6000
##          Detection Rate : 0.5550
##    Detection Prevalence : 0.5950
##       Balanced Accuracy : 0.9125
##
##        'Positive' Class : 0
##
```

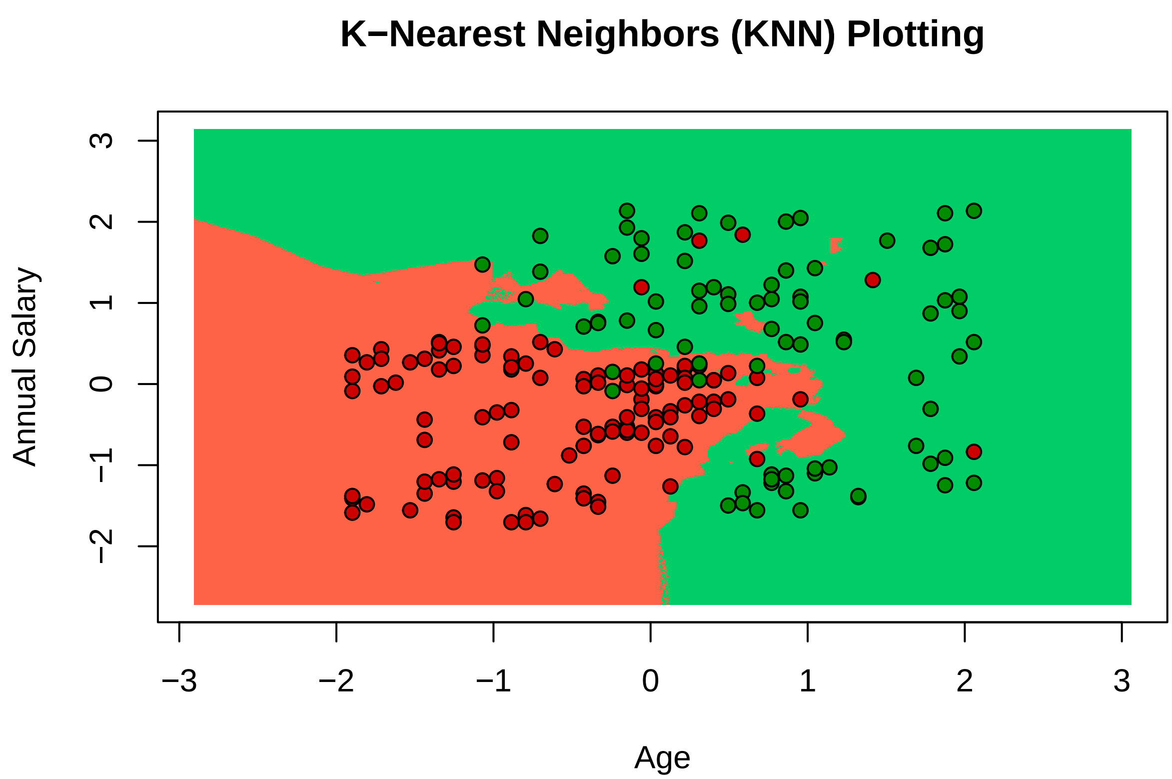


K-Nearest Neighbors Prediction

```r
knn_pred()
```

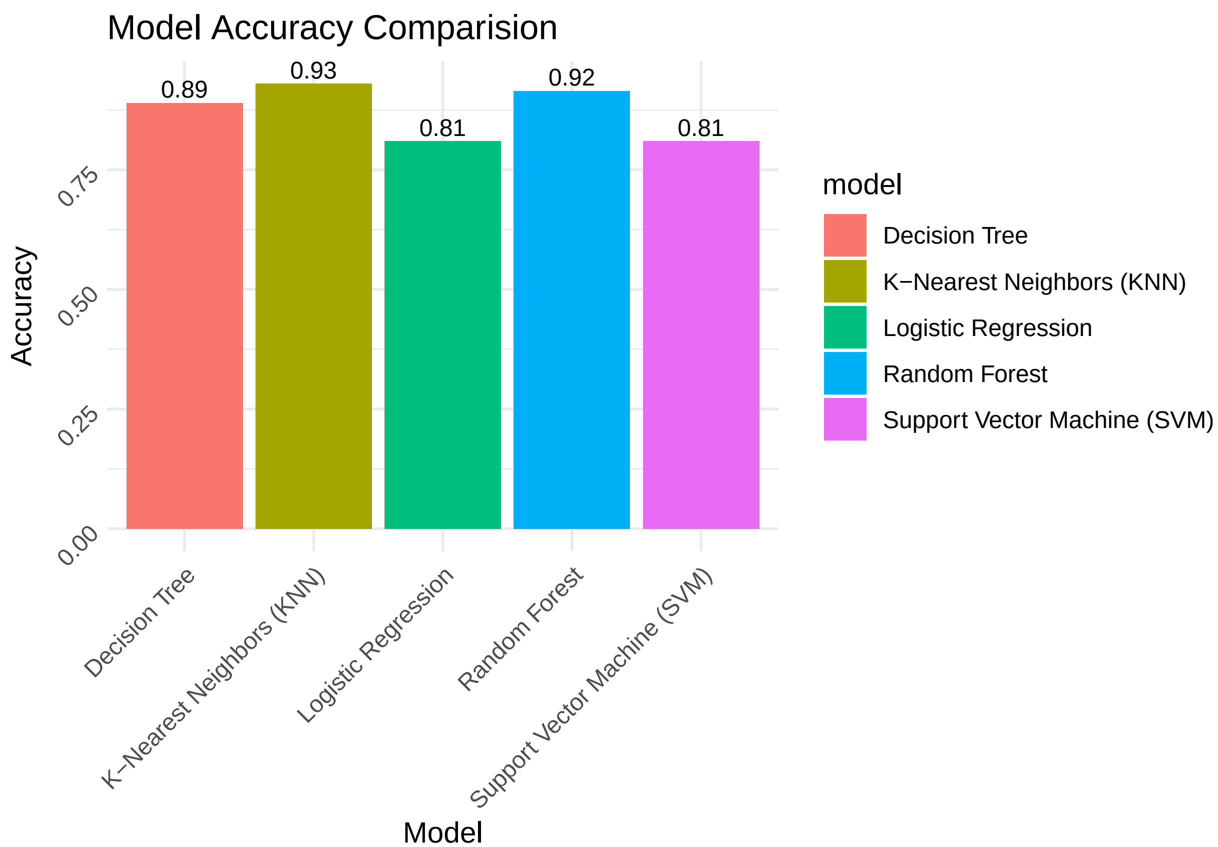
```
##
## [1] "K-Nearest Neighbors (KNN)"
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##           0 114   8
##           1   6  72
##
##                Accuracy : 0.93
##                  95% CI : (0.8853, 0.9612)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8536
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9500
##             Specificity : 0.9000
##          Pos Pred Value : 0.9344
##          Neg Pred Value : 0.9231
##              Prevalence : 0.6000
##          Detection Rate : 0.5700
##    Detection Prevalence : 0.6100
##       Balanced Accuracy : 0.9250
##
##        'Positive' Class : 0
##
```



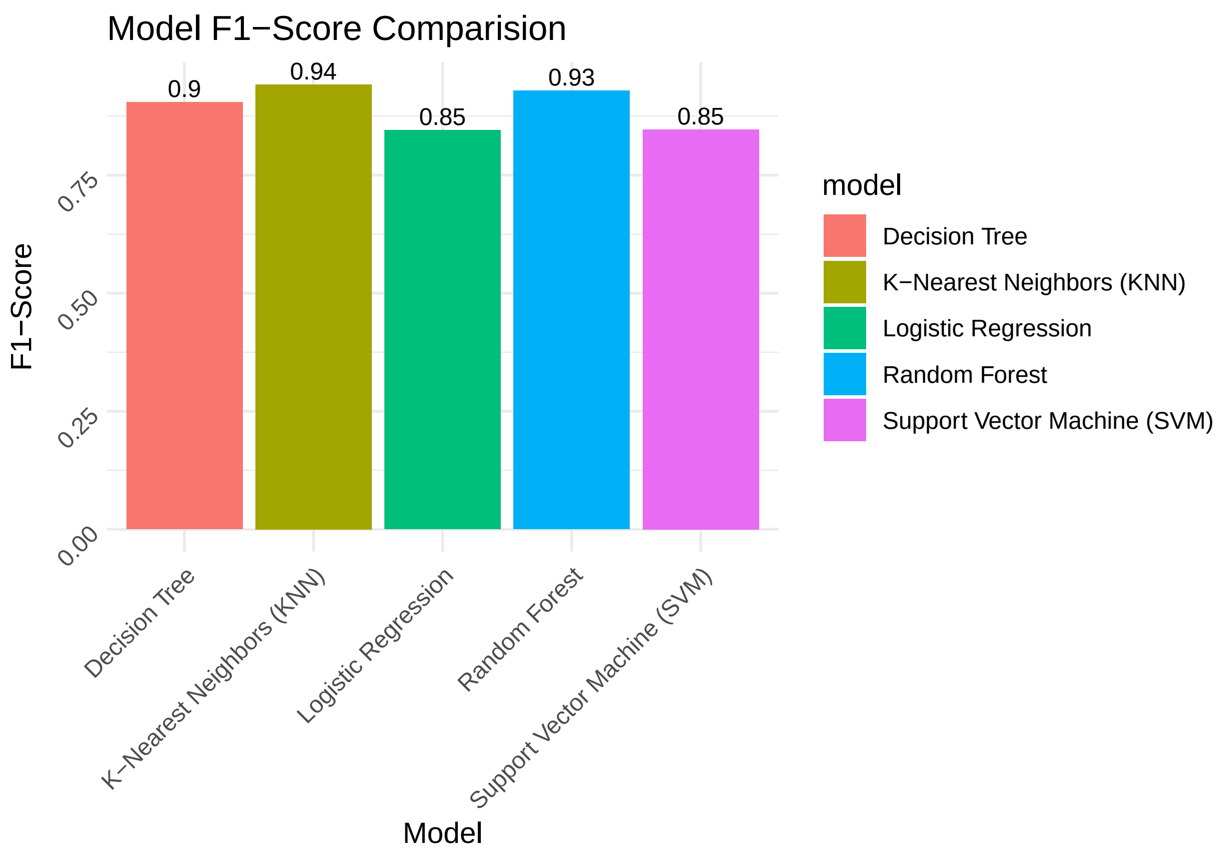
Bar Chart of Comparision of Accuracy of All Models

```r
draw_bar_chart("Accuracy")
```



Bar Chart of Comparision of F1-Score of All Models

```r
draw_bar_chart("F1-Score")
```


Model F1−Score Comparision
0.9
0.94
0.85
0.93
0.85
F1−Score
0.00
0.25
0.50
0.75
Decision Tree
K−Nearest Neighbors (KNN)
Logistic Regression
Random Forest
Support Vector Machine (SVM)
Model
model
Decision Tree
K−Nearest Neighbors (KNN)
Logistic Regression
Random Forest
Support Vector Machine (SVM)